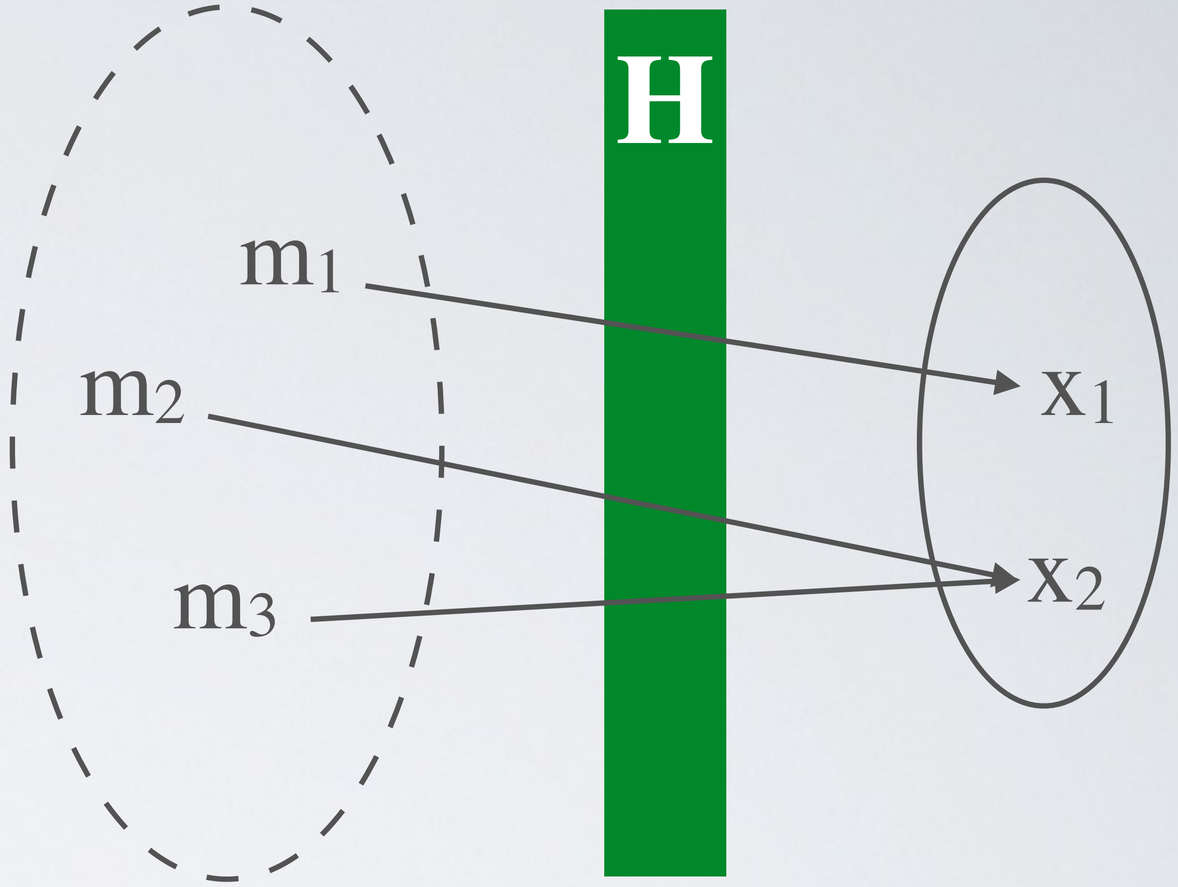
Cryptographic Hash Functions

Thierry Sans

Cryptographic hashing



$H(m^n) = m'^{n'}$ is a hash function if

- **H** is one-way function
- **n** (bit length) is unbounded
- **n'** is short (and usually fixed)

➡ **H** is a lossy compression function

Two families of hash functions

- Non-keyed a.k.a message digest
  e.g. password protection, digital signatures

  $$H(m^n) = m'^{n'}$$

- Keyed a.k.a MAC - Message Authentication Code
  e.g. message integrity

  $$H_k(m^n) = m'^{n'}$$

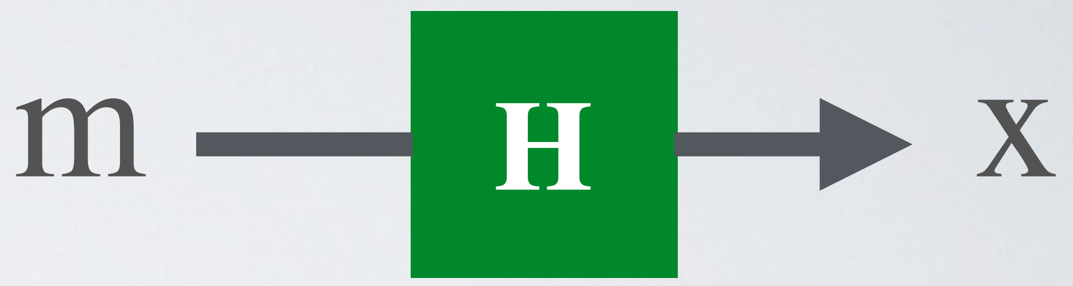
Computational complexity

$$\begin{tikzcd} m \arrow[r, dash] & \boxed{H} \arrow[r] & x \end{tikzcd}$$

- Given H and m, computing x is **easy** (polynomial or linear)
- Given H and x, computing m is **hard** (exponential)

➡ H is **not invertible**

Preimage resistance and collision resistance


$$\begin{tikzcd} m \arrow[r] & \boxed{H} \arrow[r] & x \end{tikzcd}$$

PR - Preimage Resistance

➡ given H and x, hard to find m
e.g. password storage

2PR - Second Preimage Resistance

➡ given H, m and x, hard to find m' such that H(m) = H(m') = x
e.g. virus resistance (Tripwire tool)

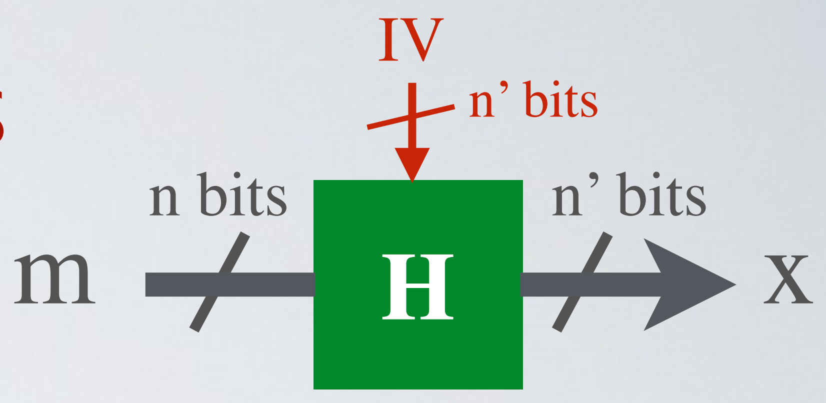
CR - Collision Resistance

➡ given H, hard to find m and m' such that H(m) = H(m') = x
e.g. digital signatures

CR ⇒ 2PR ⇒ PR

Hash functions in practice

Non-keyed vs Keyed hash functions



Most hash functions require an IV (Initialization Vector)

- **Non keyed**

  the IV (Initialization Vector) is fixed

  $H(m^n) = m'^{n'}$

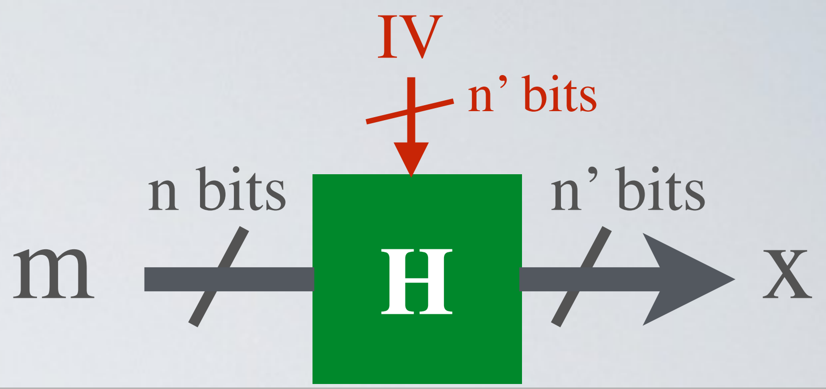
- **Keyed**

  the key is supplied as the IV

  $H_k(m^n) = m'^{n'}$

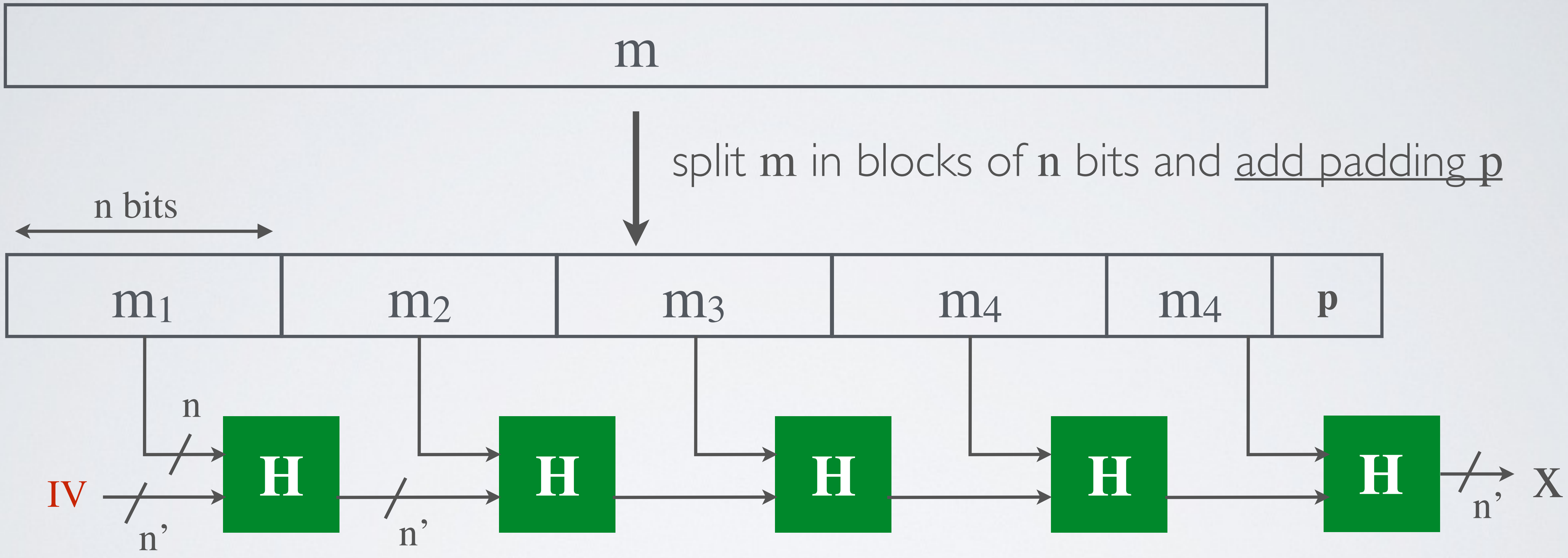
➡ The commonly used standards are non keyed

Common hash functions



Name	MD5	SHA-1	SHA-2				SHA-3			
Variant			SHA-224	SHA-256	SHA-384	SHA-512	SHA3-224	SHA3-256	SHA3-384	SHA3-512
Year	1992	1993	2001				2012			
Designer	Rivest	NSA	NSA				Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche			
Input n bits	512	512	512	512	1024	1024	1152	1088	832	576
Output n' bits	128	160	224	256	384	512	224	256	384	512
Speed cycle/byte	6.8	11.4	15.8		17.7		12.5			
Considered Broken	**yes**	**yes**	**no**				**no**			

How to hash long messages ?
Merkle–Damgård construction



Property : if H is CR then Merkel-Damgard is CR

Security of hash functions

Brute-forcing a hash function

$$\begin{tikzcd} m \arrow[r, dash] & \boxed{H} \arrow[r] & x \end{tikzcd}$$

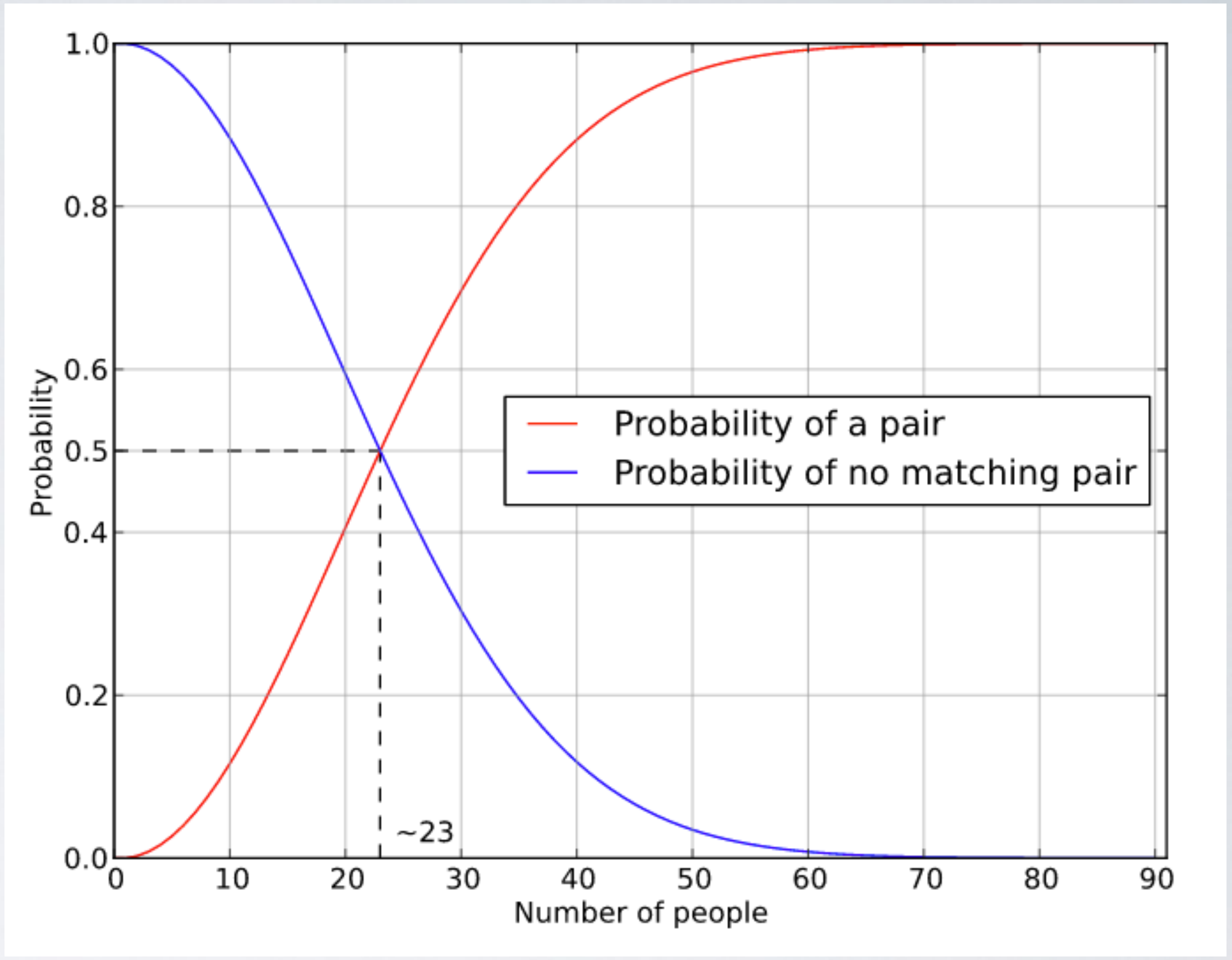
CR - Collision Resistance

➡ given H, hard to find m and m' such that $H(m) = H(m') = x$

Given a hash function H of n bits output

- Reaching all possibilities 2^n cases
- On average, an attacker should try half of them $\cancel{2^{n-1}}$ cases

Birthday Paradox



"There are 50% chance that 2 people have the same birthday in a room of 23 people"

N-bits security

➡ Given a hash function **H** of **n** bits output,

a collision can be found in around $\mathbf{2^{n/2}}$ evaluations

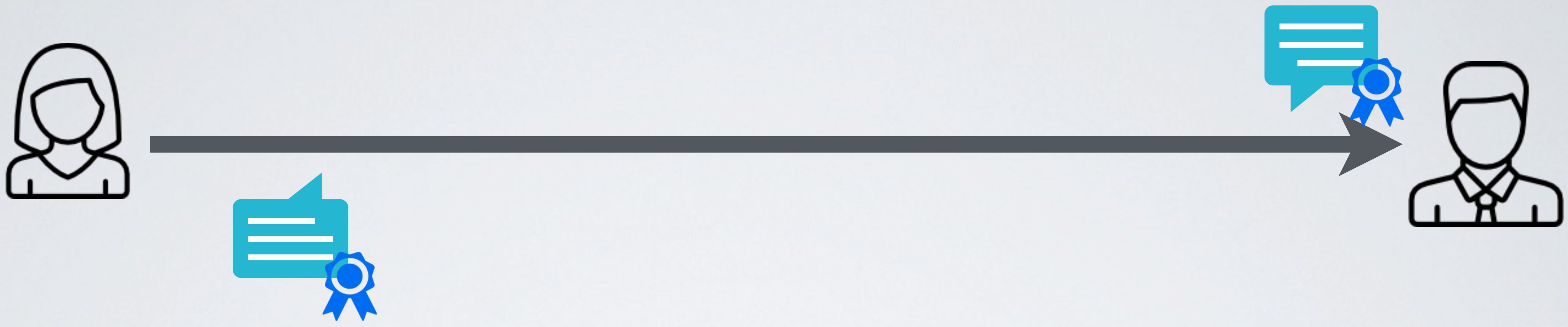
e.g SHA-256 is 128 bits security

Broken hash functions beyond the birthday paradox

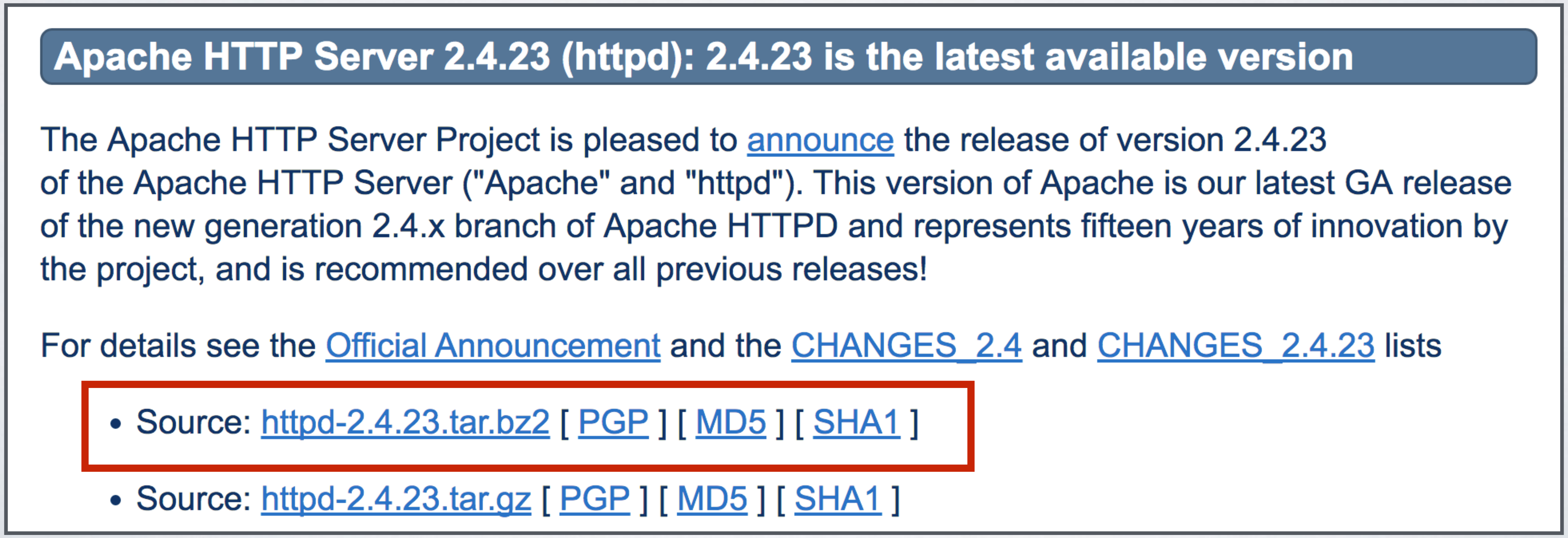
	Year	Collision
MD5	2013	2^{24} evaluations (2^{39} with prefix)
SHA-1	2015	2^{57} evaluations

Using hash functions for Integrity

Hashing



$m \,||\, H(m)$



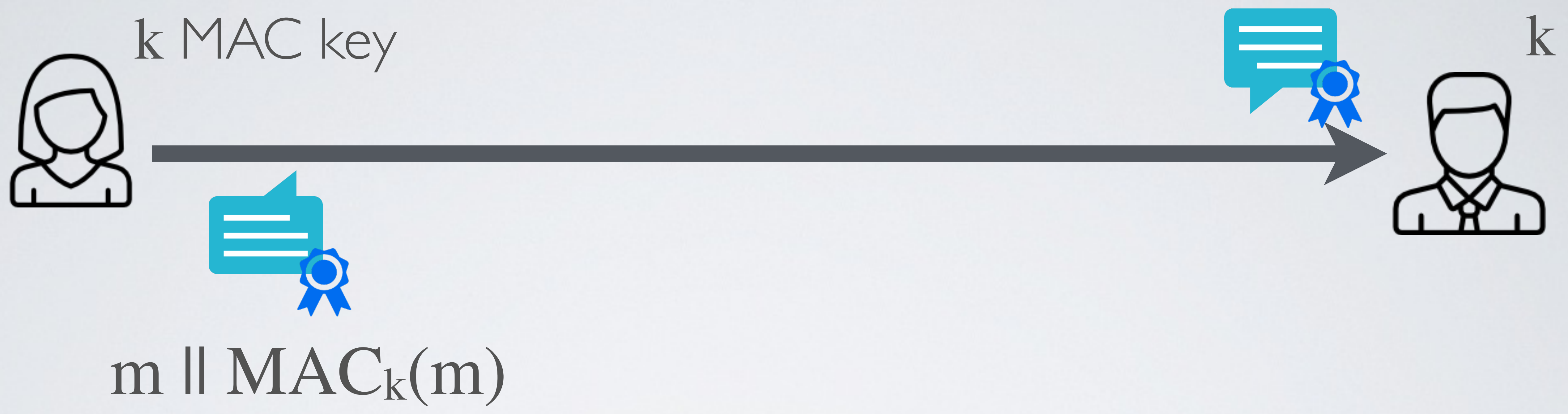
Apache HTTP Server 2.4.23 (httpd): 2.4.23 is the latest available version

The Apache HTTP Server Project is pleased to announce the release of version 2.4.23 of the Apache HTTP Server ("Apache" and "httpd"). This version of Apache is our latest GA release of the new generation 2.4.x branch of Apache HTTPD and represents fifteen years of innovation by the project, and is recommended over all previous releases!

For details see the Official Announcement and the CHANGES_2.4 and CHANGES_2.4.23 lists

- Source: httpd-2.4.23.tar.bz2 [PGP] [MD5] [SHA1]
- Source: httpd-2.4.23.tar.gz [PGP] [MD5] [SHA1]

MAC - Message Authentication Code



Alice an Bob share a key **k**

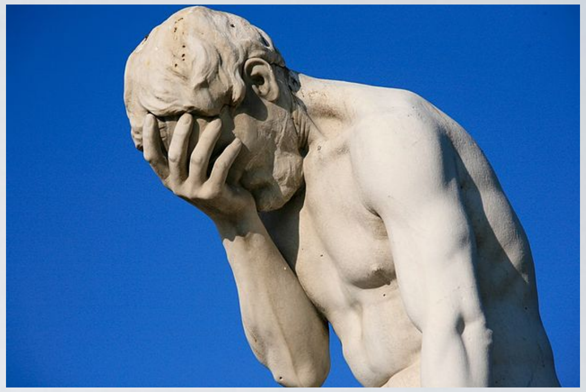
➡ Option 1 : using a <u>keyed hash</u> function on the message

$$MAC_k(m) = H_k(m)$$

➡ Option 2 : using a <u>non-keyed</u> hash function on the message (HMAC)

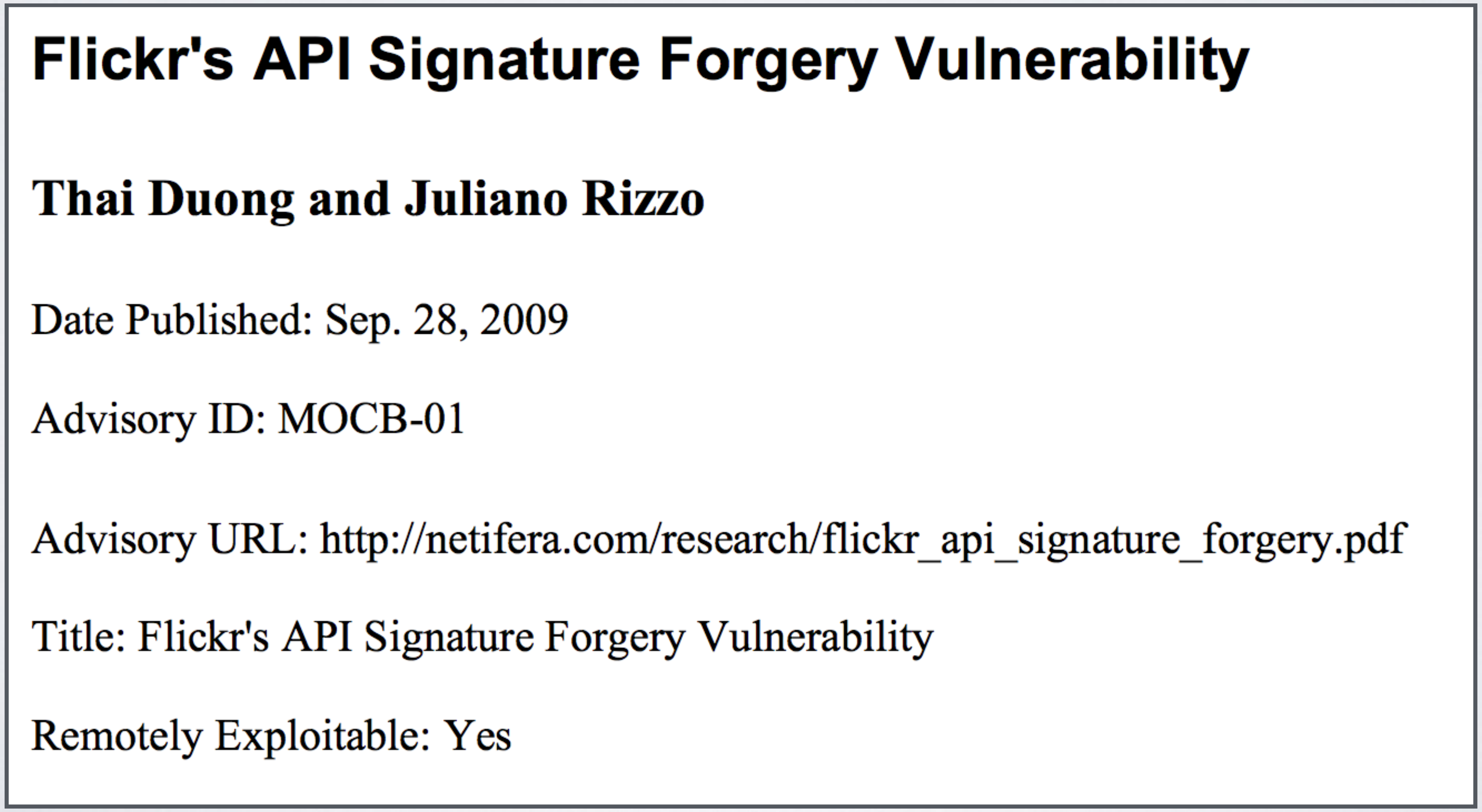
$$MAC_k(m) = H(\cancel{k \,||\, m})$$

Length extension attack



$$MAC_k(m \| m') = H(MAC_k(m) \| m')$$

Vulnerable : MD5, SHA-1 and SHA-2 (but not SHA-3)



Flickr's API Signature Forgery Vulnerability

Thai Duong and Juliano Rizzo

Date Published: Sep. 28, 2009

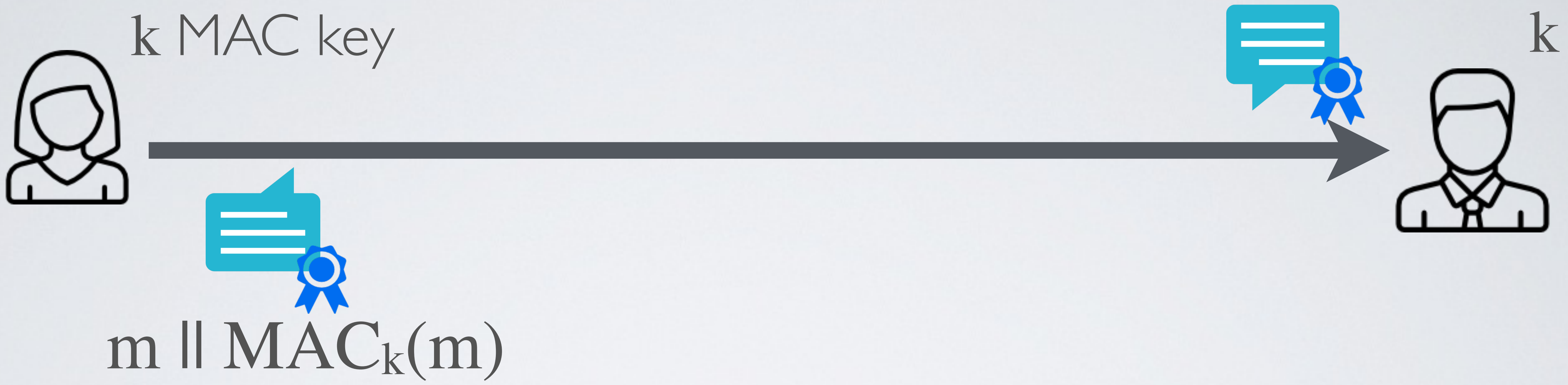
Advisory ID: MOCB-01

Advisory URL: http://netifera.com/research/flickr_api_signature_forgery.pdf

Title: Flickr's API Signature Forgery Vulnerability

Remotely Exploitable: Yes

Good MACs with non-keyed hash



Alice an Bob share a key k

➡ Option 1 : envelope method

$$MAC_k(m) = H(k \| m \| k)$$

➡ Option 2 : padding method (i.e. HMAC standard)

$$HMAC_k(m) = H((k \oplus opad) \| H((k \oplus ipad) \| m))$$